# Dokumentation von SAS Programmen mit AutoDoc

Daniel Enache
Targobank AG & Co. KG aA
Kasernenstr. 10
40213 Düsseldorf
daniel.enache@targobank.de

**Zusammenfassung**

Das Open Source Programm AutoDoc ermöglicht über spezielle Kommandos in den Kommentaren eines Programms, automatisch eine technische Dokumentation zu erstellen. So kann der Aufwand zur Erstellung einer solchen Dokumentation deutlich verringert werden. Es unterstützt fast alle Programmiersprachen und ist daher insbesondere gut für SAS-Programme geeignet.

**Schlüsselwörter:** Dokumentation, AutoDoc

## 1 Motivation

Im professionellen Umfeld ist die Dokumentation von Programmen unerläßlich. Dies könnte regulatorische Gründe haben, falls eine Pflicht zur Dokumentation besteht. Aber auch praktische Gründe sprechen für eine gute technische Dokumentation. Einerseits kann man sich dadurch nach langer Zeit rasch wieder in umfangreiche Programme hineindenken, andererseits vereinfacht sie die Übergabe von Programmen an andere Programmierer.
Der zusätzliche Aufwand für die Dokumentation sollte dabei jedoch möglichst gering gehalten werden. Daher bietet sich ein System an, das direkt Kommentare im Quellcode analysiert und daraus die Dokumentation automatisch erstellt.

Im Vortrag wurde auf die Motivation zur Verwendung von AutoDoc eingegangen, insbesondere weil andere Programme, wie Javadoc™ [2] und Doxygen™ [3] entweder gar nicht oder nur sehr eingeschränkt mit SAS-Programmen verwendet werden können. Dazu wird eine mit AutoDoc erzeugte Dokumentation im HTML-Format im Browser vorgestellt, um einen Eindruck von der Qualität und den Möglichkeiten zu erhalten. Ein Beispiel, wie eine Ausgabe im Browser aussehen könnte, wird in der folgenden Abbildung 1 dargestellt. In diesem Beitrag soll nun näher auf die Benutzung des Programmes eingegangen werden.

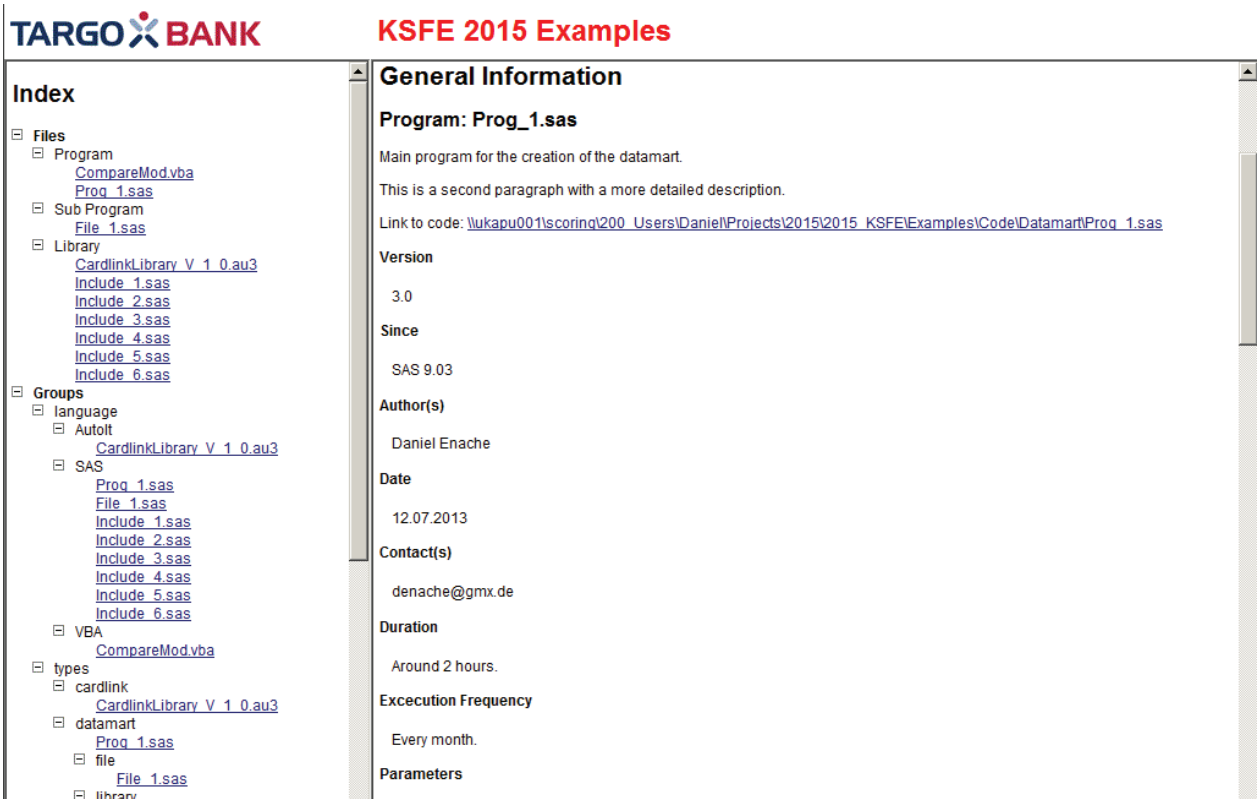


**Abbildung 1:** Beispiel Dokumentation im Browser

# 2 Markierung des Quelltextes

## 2.1 Einführung

Der erste Schritt zur automatischen Erstellung einer Dokumentation mit AutoDoc ist es, entsprechende Kommentare in den Quellcode der zu dokumentierenden Programme einzufügen. AutoDoc verwendet spezielle Kommentarblöcke, um die automatische Generierung der Dokumentation zu steuern. Diese Blöcke beginnen mit speziellen Markern und werden durch andere Marker beendet. Diese Marker sind so definiert, dass sie für die jeweilige Programmiersprache einen gültigen Kommentarblock darstellen.

Für die Programmiersprache SAS beginnen diese speziellen Kommentarblöcke standardmäßig mit `/*!` und enden mit `*/`, dem normalen Block-Kommentar-Ende von SAS. Durch spezielle Kommandozeilen-Parameter oder andere Einstellmöglichkeiten können diese Programmvorgaben geändert werden. Dies ist in der AutoDoc-Dokumentation beschrieben.

Innerhalb dieser Kommentare werden spezielle Kommandos, sogenannte *Tags*, verwendet, um Informationen zur Dokumentation hinzuzufügen. Diese Tags beginnen mit einem `@` (äquivalent zu JavaDoc). Es kann jedoch auch alternativ der Backslash (`\`) verwendet werden (äquivalent zu Doxygen), man sollte sich nur innerhalb einer Datei für eine Variante entscheiden.

AutoDoc unterscheidet 5 Arten von Tags:

- **Datei-Tags:** Beschreiben die Datei.
- **Struktur-Tags:** Beschreiben die Struktur, wie Includes, Programmabschnitte, Klassen, etc.
- **Funktionale Tags:** Beschreiben funktionale Elemente, wie Funktionen, Makros, Variablen, Formate, etc.
- **Unter-Tags:** Fügen weitere Informationen zu Datei-, Struktur- und funktionalen Tags hinzu (z.B.: Autor, Version, Datum, etc.).
- **Inline-Tags:** Können innerhalb des Beschreibungs-Textes anderer Tags verwendet werden, um spezielle Effekte und Querverweise einzufügen.

Um in die Dokumentation aufgenommen zu werden, muss eine Quelldatei einen Datei-Tag enthalten. Ein sehr einfacher Header könnte wie folgt aussehen:

```
/*!*************************************************************
* @program Dies ist das Hauptprogramm für das monatliche Reporting.
*
*          Hier kommt der zweite Absatz, der eine detaillierte
*          Beschreibung enthält.
*************************************************************/
```

Die Datei wird dadurch als „Programm“ markiert und als Kurz-Beschreibung der Text „Dies ist das Hauptprogramm für das monatliche Reporting.“ hinzugefügt. Neue Absätze können durch Leerzeilen erzeugt werden. Auf diese Weise können auch Kurzbeschreibung von detaillierter Beschreibung getrennt werden.

Es werden bei solchen detaillierteren Beschreibungen automatisch nicht nummerierte Aufzählungen (eingeleitet durch `-`) und nummerierte Aufzählungen erkannt, wenn mindestens zwei Listenelemente vorhanden sind. Wie man an diesem Beispiel sieht, ist es auch möglich, den Kommentarblock durch `*` zu rahmen, um die Übersichtlichkeit im Quelltext zu erhöhen. Ein solcher Rahmen wird von AutoDoc erkannt und automatisch ignoriert.

## 2.2 Datei-Dokumentation

Der Datei-Header enthält den wichtigen Datei-Tag, mit dem festgelegt wird, um was für eine Art von Datei es sich handelt. Es kann hier zwischen `@program`, `@subprogram`, `@library`, `@file` und `@exclude` gewählt werden (Der `@exclude` Tag ist eine Möglichkeit, eine bestimmte Datei von der Verarbeitung auszuschließen.).

Gefolgt werden diese Tags jeweils von einer Beschreibung, wie im vorigen Abschnitt beschrieben. Dann können über Unter-Tags zusätzliche Informationen, wie Autor, Datum, Version, etc., hinzugefügt werden. Hier folgt ein sehr ausführliches Beispiel:

```
/*!*************************************************************
* @program    Main program for the creation of the datamart.
*
*    This is a second paragraph with a more detailed description.
*
* @version    3.0
* @since      SAS 9.03
* @author     Daniel Enache
* @date       12.07.2013
* @contact    denache@gmx.de
* @duration   Around 2 hours.
* @freq       Every month.
*
* @param      BAUMON    Month for the BAU run in YYYYMM format. Will
*                       be set automatically by the calling routine.
* @input      TAB1yyyy       Input table 1.
* @input      TAB2yyyy       Input table 2.
* @output     OUT_TAB1yyyy   Output table 1.
*
* @precond   The following tables must be present:
*            - TAB1yyyy
*            - TAB2yyyy
*
* @group      types/datamart
*
* @howto      1. Set the macro variable paramter BAUMON to the last
*               complete month.
*            2. Run the program.
*
*            Now, these are the steps to check the log file:
*            3. Open the log file in the editor.
*            4. Search for ERROR's.
*
*            Finished.
*
* @history VERS DATE        AUTHOR           DESCRIPTION
* @history 3.0  01.01.2013 "Daniel Enache"  Initial implementation.
* @history 3.1  12.07.2013 "Daniel Enache"  - Corrections.
*                                           - Other changes.
*                                           - Some more changes.
*************************************************************/
```

Dieses Beispiel demonstriert das Zusammenspiel zwischen Haupt-Tags, wie dem Datei-Tag, und den Unter-Tags, sowie neue Absätze und Aufzählungen. Dieser Datei-Header wurde beispielsweise verwendet, um die Ausgabe aus der letzten Abbildung zu erhalten. Tag-Parameter, die Leerzeichen enthalten, müssen in Anführungszeichen gesetzt werden müssen, damit AutoDoc diese nicht trennt. Der letzte Parameter eines jeden Tags ist immer eine Beschreibung und hier kann auf Anführungszeichen verzichtet werden. Daher ist im letzten Beispiel im `@history` Tag der Name des Autors in Anführungszeichen gesetzt, die Beschreibung jedoch nicht.

## 2.3 Dokumentation der Struktur

Für SAS sind vor allem die Struktur-Tags `@progsection`, `@step` und `@includes` interessant. Mit `@progsection` und `@step` kann ein Programm strukturiert werden und mit dem `@includes` Tag können inkludierte Dateien markiert werden. Auf diese Weise kann bei komplexeren Programmsystemen ein kompletter Include-Baum generiert werden, um so die komplette Programmstruktur erfassen zu können.

Ein Beispiel könnte wie folgt aussehen:

```
/*!
* @includes "/some/network/path/Signon.sas"   Standard login to DWH.
* @includes "/some/network/path/DWH_ZipUnzip_Lib.sas" Import Zip-
*                                                    Library.
*/
```

*(SAS Code entfernt)*

```
/*!*************************************************************
* @progsection Include Macro Libraries.
* @includes "./include/Include_1.sas" Utility macros.
* @includes "include/Include_2.sas"   Macros for creation of a
*                                     {@i special} type of tables.
* @includes "include\..\include\Include_3.sas" Macros for creation
*                                              of other tables.
* @includes ".\Include\Include_4.sas" Macros for creation of yet
*                                     other data.
* @includes "include/Include_5.sas"   Macros for core routines of
*                                     the program.
**************************************************************/
```

*(SAS Code entfernt)*

```
/*!*************************************************************
* @progsection Main program.
**************************************************************/
```

*(SAS Code entfernt)*

Die Ausgabe im Browser ist in Abbildung 2 dargestellt.

Man beachte, dass Includes auf verschiedene Weise relativ angegeben werden können. Findet AutoDoc diese in den angegebenen Eingabeverzeichnissen, dann werden Links zu der Dokumentation dieser Dateien generiert und ein kompletter Include-Baum erzeugt.

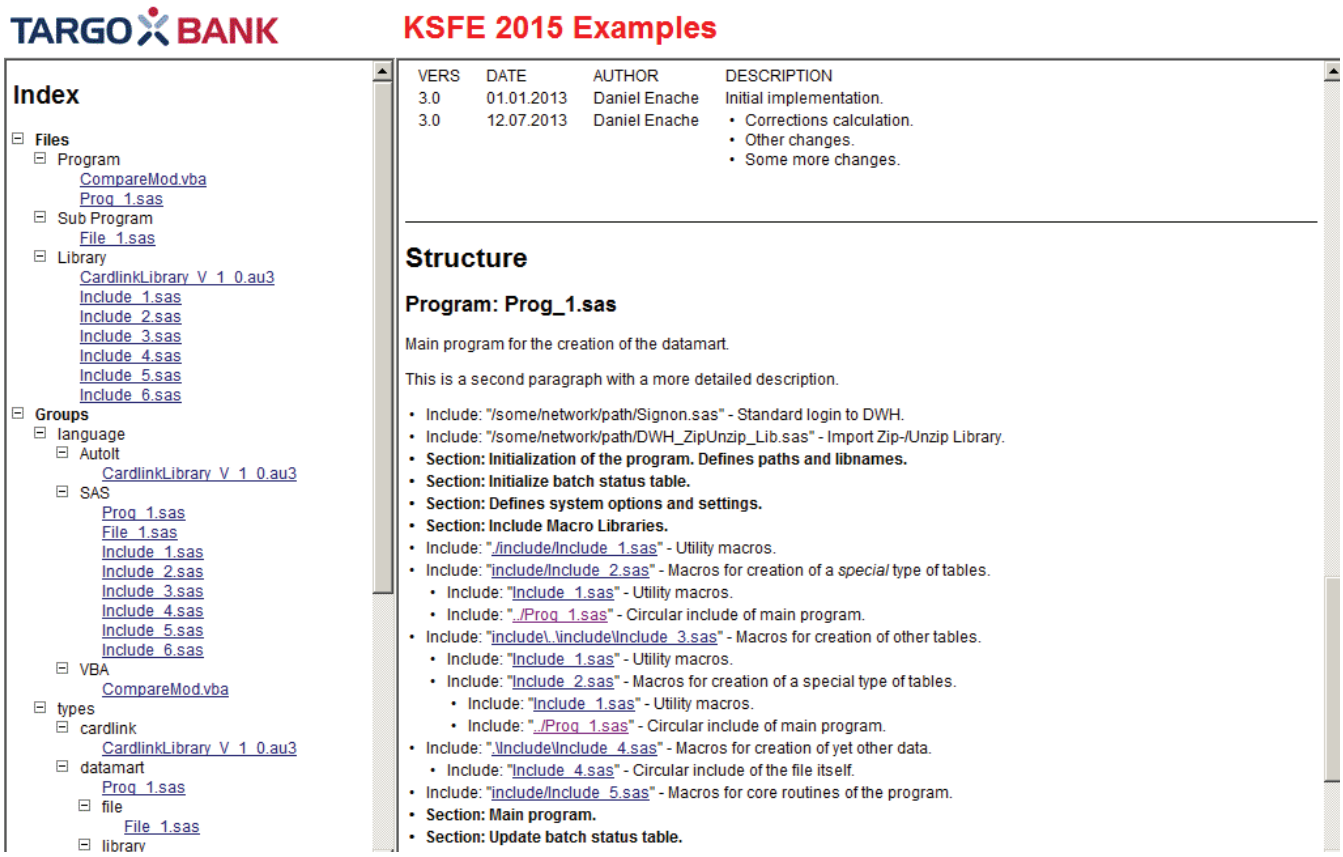


**Abbildung 2:** Beispiel Struktur-Tags im Browser

## 2.4 Dokumentation von funktionalen Elementen

Für SAS Programmierer sind vor allem die Tags `@macro`, `@function`, `@format` und `@variable` interessant. Auch diese können mit Unter-Tags kombiniert werden, um zusätzliche Informationen hinzuzufügen.

Ein Beispiel für eine Makro-Beschreibung könnte so aussehen:

```
/*!*************************************************************
* @macro "setGroup(lib, table, gr)"
*        Sets the user group of a SAS dataset file (Unix only).
*
* @param   lib     Libname.
* @param   table   Table name.
* @param   gr      Group name (see Unix command "chgrp").
*************************************************************/
%macro setGroup(lib, table, gr);
...
```

Man beachte, dass der erste Parameter des `@macro` Tags Leerzeichen enthält und daher in Anführungszeichen eingeschlossen ist. Der `@param` Unter-Tag ermöglicht die Dokumentation der Parameter, die dem Makro übergeben werden. Bei funktionalen Makros können auch Rückgabewerte dokumentiert werden, wie im folgenden Beispiel:

```
/*!*************************************************************
* @macro "getRows(table)"
*        Gets the number of rows in a SAS table.
*
* @param    table   SAS table name.
* @return   Resulting row number, or missing, if table could not be
*           opened.
**************************************************************/
%macro getRows(table);
...
```

Über den Unter-Tag `@calls` können andere Makros dokumentiert werden, die von diesem Makro aufgerufen werden:

```
/*!*************************************************************
* @macro "MAIN_TABLE_SAVE(LOOKUPYEAR, ZIP)"
*        Stores the type 1 file into the directory of the
*        datamart.
*
* @param  LOOKUPYEAR   Current lookup year.
* @param  ZIP          Should file be zipped. (Y=yes, N=No)
* @calls  "ziplib_zip(fileref)"
* @calls  "getPath(fileref)"
**************************************************************/
%macro MAIN_TABLE_SAVE(LOOKUPYEAR, ZIP);
...
```

Wenn die so referenzierten Makros in den untersuchten Eingabeverzeichnissen gefunden werden, dann wird automatisch ein Link zu jeweiligen Dokumentation des Makros erzeugt.

Der Tag `@function` kann dazu verwendet werden, FCMP Funktionen zu dokumentieren, wie im folgenden Beispiel:

```
/*!*************************************************************
* @function "xlogx(x, base)"
*    Calculates x * log(x) for a given log base.
*
*    This is a helper function, since log(0) is undefined and would
*    lead to error messages in SAS.
* @param    x     Argument.
* @param    base  Base for logarithm.
* @return   Calculated x * log(x).
**************************************************************/
proc fcmp;
...
```

Variablen und Formate sind streng genommen keine Funktionen, werden in AutoDoc jedoch ähnlich behandelt und daher finden sich die Tags `@variable` und `@format` ebenfalls unter den funktionalen Tags.

```
/*!*************************************************************
* @variable   BAUMON    Reporting month in {@code yyyymm}
*                       format (global).
**************************************************************/
%let BAUMON=201304;
...

/*!*************************************************************
* @format  BIN
*          Format for the binning of a probability.
**************************************************************/
proc format;
    value BIN (notsorted)
    ...
run;
```

Variablen und Formate werden in einer Übersicht zusammen, jedoch getrennt von Funktionen und Makros ausgegeben.

## 2.5 Spezielle Tags (Inline-Tags)

Inline-Tags befinden sich in geschweiften Klammern und können in der Beschreibung anderer Tags auftauchen, um bestimmte Effekte zu erzielen oder Querverweise zu ermöglichen. Sie können jedoch nicht verschachtelt werden. Externe Links, z.B. zu Webseiten, können mit dem Tag `@extlink` hinzugefügt werden, wie im folgenden Beispiel:

```
/*!*************************************************************
* @library  Contains some utility macros.
*
*           For additional information see
*           {@extlink "http://www.google.de/" Google Suchseite}.
**************************************************************/
```

Interne Querverweise können über den Tag @link hinzugefügt werden:

```
/*!*************************************************************
* @macro "initMainTable(LOOKUPYEAR)"
*        Initializes the type 1 tables of the datamart.
*        Makes sure filed names, field widths and lengths are
*        correct.
*
* @param  LOOKUPYEAR   Current lookup year.
*
* @see {@link "Include_2.sas#INIT_TABLE2(TTDMON)" INIT_TABLE2 in
*      file Include_2.sas}. Klick on the link to see the
*      documentation.
**************************************************************/
```

Der Verweis besteht aus dem Dateinamen, der das Element enthält, einem # Zeichen als Trennsymbol und der Funktions- bzw. Makrodefinition auf des zu verweisenden funktionalen Elements. Will man nur auf die Datei verweisen, dann lässt man den letzten Teil

weg, jedoch nicht das Trennsymbol #, z.B.: `@see {@link "Include_2.sas#" File Include_2.sas}` Soll auf ein funktionales Element innerhalb derselben Datei verwiesen werden, dann kann man den Dateinamen und das Trennsymbol weglassen, wie im folgenden Beispiel:

```
 @see {@link "INIT_TABLE2(TTDMON)" INIT_TABLE2 in the current file}
```

Inline-Tags können auch dazu verwendet werden, Texthervorhebungen zu erreichen, ohne auf HTML zugreifen zu müssen. Um zum Beispiel das Wort „main“ fett zu drucken, kann man den Tag `@b` verwenden:

```
/*!************************************************************
* @program This is the {@b main} program.
*************************************************************/
```

# 3 Installation und Benutzung

## 3.1 Installation

AutoDoc ist ein Open-Source-Programm unter der Lizenz GPL 3.0 und kann unter [1] heruntergeladen werden. Benötigt wird eine bereits installierte Java-Runtime-Umgebung, Version 1.4 oder höher. Die Schritte im Einzelnen:

1. Herunterladen von AutoDoc_*Version*_Program.zip
2. Entpacken in ein beliebiges Verzeichnis

Ein Installations-Programm für Microsoft Windows™ ist seit Version 1.0.16 ebenfalls vorhanden, jedoch nicht für andere Betriebssysteme.
Die Programmdokumentation findet man im Unterordner „Documentation“. Dort findet man die Dokumentation im HTML-Format im Unterordner „html“ und im Windows™-Hilfe-Format im Unterordner „chm“.

## 3.2 Benutzung Grafische Benutzeroberfläche (GUI)

Seit Version 1.0.15 existiert für AutoDoc eine graphische Benutzeroberfläche (GUI), die sich momentan noch im Beta-Stadium befindet (s. Abbildung 3).

Der Aufruf erfolgt durch:

```
javaw -jar AutoDocGUI_1_0.jar
```

Im Hauptfenster können die einzelnen Parameter bequem eingestellt werden, die aktuelle Konfiguration gespeichert und wieder geladen sowie der Verarbeitungslauf gestartet werden. Eine nähere Dokumentation der Oberfläche findet man in der Programmdokumentation.

Nachdem der Verarbeitungslauf beendet ist, kann die Log-Datei gespeichert werden und die fertige Dokumentation im System-Browser geöffnet werden.

Zu beachten ist, dass AutoDoc die angegeben Eingabeverzeichnisse immer komplett mitsamt aller Unterverzeichnisse verarbeitet und das Ausgabeverzeichnis, falls vorhanden, immer erst leert, um alte Dateien aus vorherigen Läufen loszuwerden.

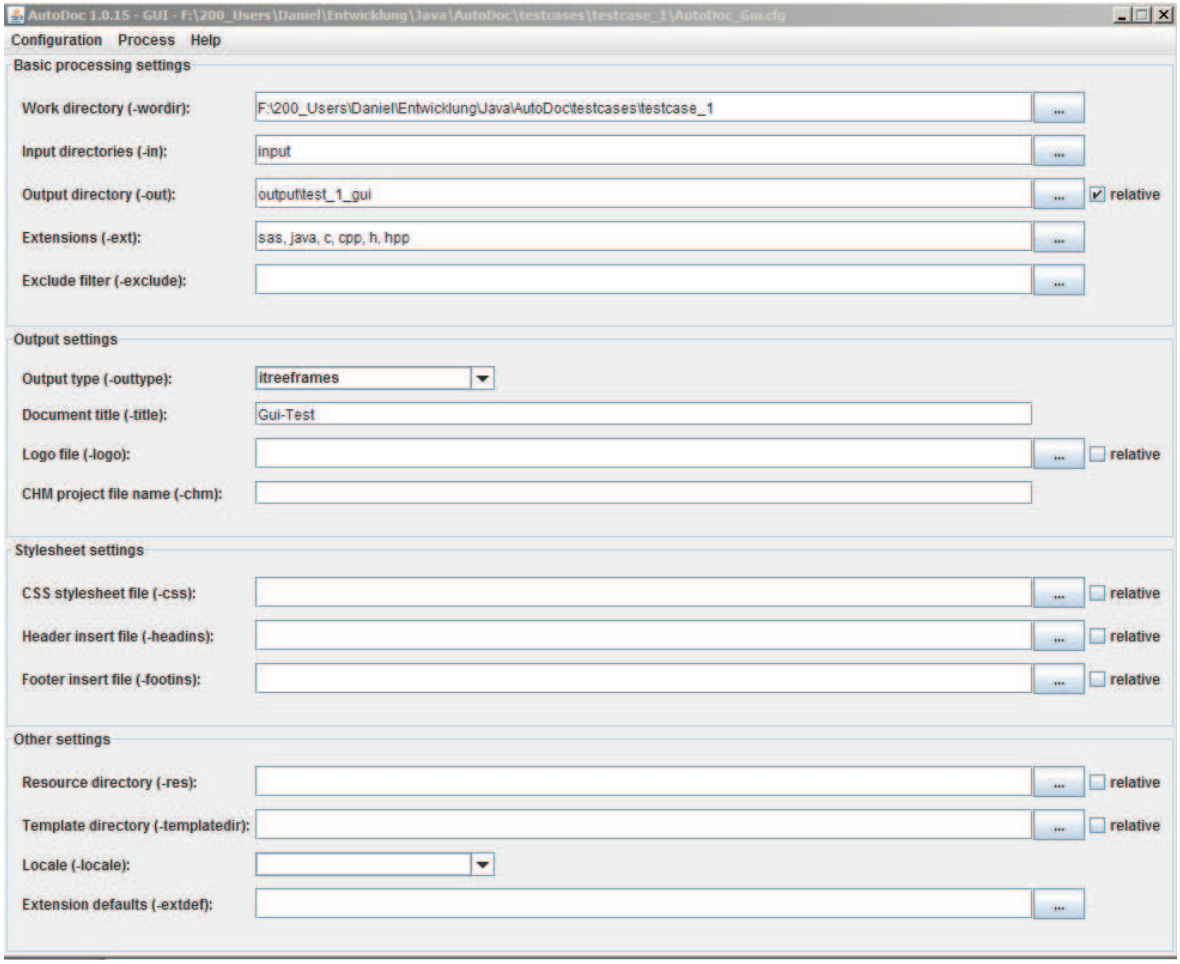


**Abbildung 3:** Hauptfenster der grafischen Benutzeroberfläche (GUI)

## 3.3 Benutzung Kommandozeile

Die Kommandozeilen-Version von AutoDoc führt man mit

```
java -jar AutoDoc_1_0.jar
```

aus. Da zumindest Eingabe- und Ausgabeverzeichnisse angegeben werden müssen, erscheint bei einem solchen Aufruf die Syntax-Beschreibung. In der Dokumentation sind die einzelnen Parameter beschrieben und Beispiele aufgeführt.

Ein einfaches Beispiel für den Aufruf mit absoluten Verzeichnissen ist:

```
java -jar AutoDoc_1_0.jar -in "C:\Temp\src" -out "C:\Temp\html"
      -title "Meine Doku"
```

Alle Programme im Eingabeverzeichnis `C:\Temp\src` inklusive aller Unterverzeichnisse werden verarbeitet und die Dokumentation in das Ausgabeverzeichnis `C:\Temp\html` erzeugt. Wenn man relative Verzeichnisse verwendet, interpretiert AutoDoc diese immer relativ zum aktuellen Arbeitsverzeichnis des Betriebssystems.

```
cd C:\Temp
java -jar AutoDoc_1_0.jar –in "src" –out "html" –title "Meine Doku"
```

wäre also Äquivalent zum obigen Beispiel.

## 3.4 Konvertierung in andere Formate

### 3.4.1 Microsoft Word™

Es ist relativ einfach, eine von AutoDoc erstellte Dokumentation in Microsoft Word™ zu konvertieren. Dazu verwendet man den Parameter `-outtype "single"`, damit die Dokumentation in einer einzigen HTML-Datei geschrieben wird. Dann öffnet man die erzeugte Datei "index.htm" mit Microsoft Word™ und speichert das Dokument als Word Dokument.

### 3.4.2 Microsoft™ Hilfe Format (CHM)

Es ist auch möglich, eine Microsoft™ Hilfe Datei (CHM) zu erstellen. Dazu benötigt man zusätzlich das Programm Microsoft Help Wokshop™, das bei Microsoft™ kostenlos bezogen werden kann.

Beim Erzeugen der Dokumentation verwendet man den Parameter `-chm "Projektname"`, damit AutoDoc zusätzlich zu der normalen Dokumentation die Hilfsdateien, z.B. "Projektname.hhp", für den Hilfe-Generator erzeugt. Die *.hpp Datei kann dann direkt mit Microsoft Help Workshop™ geöffnet und zu einer Hilfedatei kompiliert werden.

#### Literatur

[1] AutoDoc zu beziehen unter: http://sourceforge.net/projects/de-autodoc.

[2] JavaDoc™ zu beziehen im Java Development Kit (JDK): http://www.oracle.com/technetwork/java/javase/downloads/index.html.

[3] Doxygen™ zu beziehen unter: http://www.doxygen.org.

# Anhang A Befehlsreferenz

In diesem Abschnitt befindet sich die die Befehlsreferenz aller AutoDoc Tags. Sie ist der Programmdokumentation entnommen und daher englisch. Mehr Informationen findet man in der Programmdokumentation.

**Tabelle 1:** Datei Tags

| File Tag | Description | Para | Syntax |
|---|---|---|---|
| @exclude | Marks the file to be excluded from documentation. | 1 | @exclude *Description* |
| @file | Defines the file as a general file. | 1 | @file *Description* |
| @library | Defines the file as a library. | 1 | @library *Description* |
| @program | Defines the file as a program. | 1 | @program *Description* |
| @subprogram | Defines the file as a sub program which is part of a bigger program. | 1 | @subprogram *Description* |

**Tabelle 2:** Struktur Tags

| Structural Tag | Description | Para | Syntax |
|---|---|---|---|
| @class | Marks the beginning of a class. | 2 | @class Classname Description |
| @includes | Describes included files. | 2 | @includes Filename Description |
| @interface | Marks the beginning of an interface. | 2 | @interface Interfacename Description |
| @progsection | Marks the beginning of a logical section. | 1 | @progsection Description |
| @step | Describes a step (e.g. a SAS step or proc). | 1 | @step Stepname Description |

**Tabelle 3:** Funktionale Tags

| Functional Tag | Description | Para | Syntax |
|---|---|---|---|
| @field | Describes a class field. | 2 | @field *Fieldname Description* |
| @format | Describes a format definition. | 2 | @format *Formatname Description* |
| @function | Describes a function. | 2 | @function *Functionname Description* |
| @macro | Describes a macro. | 2 | @macro *Macroname Description* |
| @method | Describes a class method. | 2 | @method *Methodname Description* |
| @procedure | Describes a procedure. | 2 | @procedure *Functionname Description* |
| @variable | Describes a variable. | 2 | @variable *Variablename Description* |

**Tabelle 4:** Sub-Tags

| Sub-Tag | Description | Para | Syntax |
|---|---|---|---|
| @attention | Adds attention information. | 1 | @attention *Description* |
| @author | Adds author information. | 1 | @author *Authorname* |
| @brief | Adds brief description (for Doxygen™ compatibility only). | 1 | @brief *Description* |
| @bug | Adds bug information. | 1 | @bug *Description* |
| @calls | Adds function, method, or macro which is called. Experimental. | 1 | @calls *RoutineName* |
| @comment | Suppresses output of the following parameter. | 1 | @comment *Suppressed* |
| @contact | Adds contact information. | 1 | @contact *Contactinfo* |
| @copyright | Adds copyright information. | 1 | @copyright *Description* |
| @date | Adds date information. | 1 | @date *Date* |

**Tabelle 5:** Sub-Tags (Fortsetzung)

| Sub-Tag | Description | Para | Syntax |
|---|---|---|---|
| @deprecated | Marks the parent tag as deprecated. | 1 | @deprecated *Description* |
| @details | Adds detailed description (for Doxygen™ compatibility only). | 1 | @details *Description* |
| @docu | Adds documentation information. | 1 | @docu *Description* |
| @duration | Adds information about the duration of the program run. | 1 | @duration *Description* |
| @example | Adds an example. | 1 | @example *Description* |
| @extends | Adds information about the parent class. | 1 | @extends *Description* |
| @freq | Adds information about the excecution frequency of the program. | 1 | @freq *Description* |
| @group | Adds group information. | 1 | @group *Groupname* |
| @history | Adds history log information. | 4 | @history *Version Date Author Description* |
| @howto | Adds usage information. | 1 | @howto *Description* |
| @implements | Adds information about the implemented interface. | 1 | @implements *Description* |
| @input | Describes input data. | 2 | @input InputName Description |
| @licence | Describes the underlying software licence. | 1 | @licence Description |
| @module | Adds module information. | 1 | @module Modulename |
| @namespace | Adds namespace information. | 1 | @namespace Namespacename |
| @note | Adds a note. | 1 | @note Description |
| @outfile | Adds output file information | 1 | @outfile Description |
| @outpath | Adds output path information | 1 | @outpath Description |
| @output | Describes output data. | 2 | @output OutputName Description |
| @package | Adds package information. | 1 | @package Packagename |
| @param | Adds information about a parameter. | 2 | @param Name Description |
| @postcond | Adds post-condition information. | 1 | @postcond Description |
| @precond | Adds pre-condition information. | 1 | @precond Description |
| @return | Adds information about the return value. | 1 | @return Description |
| @section | Adds information of a user defined section. | 1 | @section Section-Heading Description |
| @see | Adds a cross reference description. | 1 | @see Reference Description |
| @since | Adds minimum compiler / interpeter version. | 1 | @since Version |
| @throws | Declares a thrown exception. | 1 | @throws Exceptionname |
| @todo | Adds TODO information. | 1 | @todo Description |
| @version | Adds version information. | 1 | @version Version |
| @warning | Adds warning information. | 1 | @warning Description |

**Tabelle 6:** Inline Tags

| Inline Tag | Description | Para | Syntax |
|---|---|---|---|
| @b | Sets the text in **bold** font. | 1 | {@b *Text*} |
| @bi | Sets the text in ***bold italic*** font. | 1 | {@bi *Text*} |
| @bis | Sets the text in ***bold italic strike-through*** font. | 1 | {@bis *Text*} |
| @biu | Sets the text in ***bold italic underline*** font. | 1 | {@biu *Text*} |

**Tabelle 7:** Inline Tags (Fortsetzung)

| Inline Tag | Description | Para | Syntax |
|---|---|---|---|
| @bs | Sets the text in ***bold strike-through*** font. | 1 | {@bs *Text*} |
| @u | Sets the text in ~~strike-through~~ font. | 1 | {@u Text} |
| @bu | Sets the text in bold underline font. | 1 | {@bu Text} |
| @code | Adds source code. | 1 | {@code Codefragment} |
| @em | Sets the text in emphasized font. | 1 | {@em Text} |
| @es | Sets the text in emphasized strike-through font. | 1 | {@es Text} |
| @eu | Sets the text in emphasized underline font. | 1 | {@eu Text} |
| @extlink | Adds an external link to another document. | 2 | {@extlink URL Label} |
| @i | Sets the text in italic font. | 1 | {@i Text} |
| @newline | Inserts a line break. | 0 | {@newline} |
| @link | Adds an internal link to another document. | 2 | {@link URL Label} |
| @s | Sets the text in strike-through font. | 1 | {@s Text} |
| @tt | Sets the text in typewriter font. | 1 | {@tt Text} |
| @ttb | Sets the text in bold typewriter font. | 1 | {@ttb Text} |
| @ttbi | Sets the text in bold italic typewriter font. | 1 | {@ttbi Text} |
| @ttbis | Sets the text in bold italic strike-through typewriter font. | 1 | {@ttbis Text} |
| @ttbiu | Sets the text in bold italic underline typewriter font. | 1 | {@ttbiu Text} |
| @ttbs | Sets the text in bold strike-through typewriter font. | 1 | {@ttbs Text} |
| @ttbu | Sets the text in bold underline typewriter font. | 1 | {@ttbuText} |
| @tti | Sets the text in italic typewriter font. | 1 | {@tti Text} |
| @ttis | Sets the text in italic typewriter strike-through font. | 1 | {@ttis Text} |
| @ttiu | Sets the text in italic typewriter underline font. | 1 | {@ttiu Text} |
| @tts | Sets the text in italic typewriter strike-through font. | 1 | {@tts Text} |
| @ttu | Sets the text in italic typewriter underline font. | 1 | {@ttu Text} |